EXHIBIT 17

ELSEVIER

Computer Communications 23 (2000) 1655–1666

computer communications

www.elsevier.com/locate/comcom

Internet card, a smart card as a true Internet node

P. Urien*

Bull CP8, 68 Route de Versailles BP45, 78431 Louveciennes Cedex, France

Abstract

We have defined a new concept named the Internet smart card. An Internet card is a device that is able to work as a true Internet node, and runs Transmission Control Protocol (TCP) client and TCP server applications (defined by Internet standards like the RFC 2068, HTTP 1.1…). A smart card is a single embedded chip including CPU and memory; the only means of communicating with the outside world is through a serial link. New communication architecture has been studied for both the terminal and the card. Through this stack a smart card shares the network resources located in the terminal. This concept has been implemented in a Java card and a Personal Computer, and the first results are presented here. Our first Internet card includes a web server and a trusted proxy, which add security features to the web connections. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Internet smart card; Transmission control protocol server applications; Network security

1. Introduction

Since 1998, Bull CP8 has worked on a smart card specifically dedicated to the Internet network [20]. The basic idea is to consider this smart card as a network computer, which is able to share the resources of the terminal to which it is connected (keyboard-screen-mouse-navigator-Internet access). Our goal is to transform a smart card into a true node of the Internet network; a smart card implements applications of the Internet world (http, electronic mail). In short, a smart card is a web server, it can be accessed from a web browser, and is able to manage several TCP connections (as a client or a server), for example the smart card will work like a trusted proxy.

A first generation (Java) Internet smart card has been developed. Our objective is to integrate these cards into the Internet community and to improve the security (authentication, integrity, privacy, and non-repudiation) required by the new services that appear in Internet networks, especially for nomad users.

We enter in the era of the ubiquitous computing. This means that more and more objects integrate a microprocessor, and have the capacity to be connected to the Internet (according to Frost and Sullivan 40% of the devices connected to Internet in 2001 will not be personal computers). A user (sometimes a mobile one) will use several types of terminals connected to the Internet: a mobile phone — the GPRS network will enable access to the Internet from GSM mobile equipment in Europe in early 2000; a game console; a TV set associated to a set top box; an organizer; a laptop.

In this context, the Internet card is used to authenticate a (mobile) user at an anonymous terminal. If necessary it manages the configuration of this terminal, which is required for the setting of a particular service. This technology constitutes a revolution of the ergonomics of the smart card and of its use through the Internet network.

2. Classical smart card

A smart card (SPOM-Self-Programmable One-chip Microcomputer) is a tamper-resistant device and was invented, at the end of the seventies by Michel Ugon [8]. The French group of bankcards CB (*Carte Bancaire*) was created in 1985 and has allowed the diffusion of 24 million

* Tel.: +33-1-39-66-4230; fax: +33-1-39-66-4545.
E-mail address: pascal.urien@bull.net (P. Urien).

Abbreviations: AMUX, APDU Multiplexor; APDU, Application Protocol Data Unit; CSL, Card Smart Layer; DES, Data Encryption Standard; DLL, Dynamic Linked Library; EAL, Evaluation Assurance Level; EEPROM, Electrically Erasable Programmable Read Only Memory; GSM, Global System for Mobile communications; GPRS, General Packet Radio Service; HSL, Host Smart Layer; HTTP, Hyper Text Transfer Protocol; IDEA, International Data Encryption Algorithm; IP, Internet Protocol; ITSEC, Information Technology Secure Evaluation Criteria; MIPS, Million Instructions Per Second; OS, Operating System; OSI, Open Systems Interconnections; PDU, Protocol Data Unit; RAM, Random Access Memory; RISC, Reduced Instruction Set Computer; RFC, Request For Comment; ROM, Read Only Memory; RSA, Rivest-Shamir-Adleman algorithm; SAP, Service Access Point; SHA, Secure Hash Algorithm; SL, Smart Layer; SmartTP, Smart Transfer Protocol; SSL, Secure Socket Layer; TCP, Transmission Control Protocol; USB, Universal Serial Bus

0140-3664/00/$ - see front matter © 2000 Elsevier Science B.V. All rights reserved.
PII: S0140-3664(00)00252-8

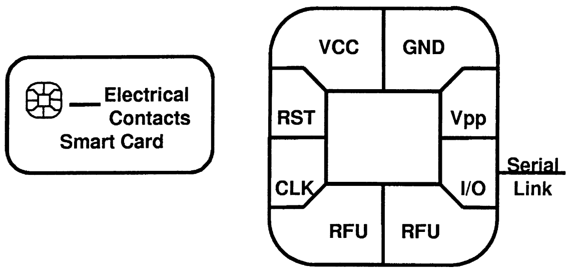

Fig. 1. Smart card physical characteristics.

devices (banking card [12]). Nowadays, smart cards are used as an electronic purse, in transport applications (contactless card [2]); in mobile phones (SIM card [3,4]); in the field of Health (French Vital Card); and for network security purposes (RSA card). One estimates a market of one billion cards including a microprocessor for the year 2000.

Today, card technology [15,22] utilizes 8 bit processors (mainly of the 6805 or 8051 family) whose memory sizes are about a few tens of kilobytes (ROM, EEPROM …). The only mean of communication with the outside world is through a serial link (see Fig. 1). As the chip has a single bi-directional I/O pin, this link can only support half-duplex protocol. The size of the chips is limited (# 25 mm^2 [12]) by the flexibility of the plastic support (PVC). A new generation of 32 bits RISC processor [21] will be available this year, with a performance of 33 MIPS, significantly higher than the performances of the traditional 8 bit micro-controller (approximately 0.6 MIPS).

The majority of chips work at the speed of 9600 baud, although the ISO standard 7816 has defined a maximum data rate of 230400 baud. A new type of SPOM, named ISO/USB, has been introduced in 1999; it provides a direct connection between a SPOM and a terminal via an USB port. This implies that the *form factor* associated to a SPOM, which is usually a plastic card, will be modified (for example key-ring) for new applications. According to USB specifications, a data throughput from 1.2 to 12 Mbit/s may be obtained, between the chip and the terminal.

A smart card is an intrinsically secure device. It is a safe place to store information or to perform processes [9,11]. Most of the attacks against this device are classified as class 3 attacks, which means that they are conducted by funded organizations that are able to support expensive and long researches; in most cases there are easier ways to get information stored in the smart card. Attacks against smart cards can be classified as different types: *logical breaking* of the micro-controller (exploiting hardware defaults [23]); *physical attacks* (chip reverse engineering [23]); and *sophisticated analysis techniques*, such as differential power analysis [24].

Security in smart card is a result of the combination of several factors that include chip design and operating system. Usually chips implement environment sensors, which detect abnormal power supply operating conditions, wrong clock frequency, or violations of chip integrity. An internal firewall can prevent data stored in the memory area (ROM, EEPROM) from being dumped by an unauthorized program. Memory addresses can be scrambled with respect to their logical addresses. A built-in timer with interrupt capability and an *Unpredictable Number Generator* can be used to impose unpredictable variations on software execution behavior, with consequent changes in the pattern of power consumption. This greatly decreases the possibility of DPA attacks. Furthermore, a Protection Profile document (*Smartcard Integrated Circuit Protection Profile*), produced by a group of integrated circuit manufacturers under the French IT Security and Certification Scheme, specifies functional and assurance requirements applicable to a smart card integrated circuit.

A smart card operating system [21,22] is organized with a machine-interface in the form of a serial link. It requires a memory capacity of few kilobytes, manages security hardware based on single chip micro-controllers, guarantees secure data storage (protected against reading or writing operations by various methods of authentication), and supports cryptographic algorithms and the associated keys (SHA1, simple to triple DES 8/16/24 byte key length, IDEA 16/32/48 byte key length, RSA with modular exponentiation 512/768/1024/2048 key length) for performing security functions (authentication, enciphering/deciphering data …). Security insurance of hardware and software development is done according to Common Criteria [25] (with for example EAL5 as a basic target [21]), which use formal methods.

2.1. Transmission protocols

As we previously mentioned, a smart card is connected to the outside world through a serial (half duplex) link, whose baud rate is between 9600 and 230400. The ISO 7816-3 standard [1] has specified two types of transmission protocol between the reader and the card.

- $T = 0$ protocol is character oriented, bytes are transmitted one at a time, and an error is detected according to a parity bit.
- $T = 1$ protocol is block oriented, bytes are transmitted in frames, and data integrity is checked by a CRC.

The ISO 7818-3 transmission protocol is equivalent to an OSI data link layer (layer 2), because it is responsible for error detection and recovery.

2.2. Application protocol data unit (APDU)

The OSI layer 5(session) and 7 (application) have been combined as a unique entity defined by the ISO 7816-4 standard. An application located in a card communicates with the outside world by means of application protocol data units (APDUs), which are exchanged through the serial link between the card and the reader. An APDU contains either a command or response messages sent from the reader to the card or vice versa.

The dialogue is carried out according to a command/response paradigm; the terminal sends a question and the card answers. In fact, the layer ISO 7816-4 defines the equivalent of an OSI session since it fixes the rules for the dialogue between the card and the reader.

An APDU command includes at least four bytes *CLA INS P1 P2*. The first two bytes (CLAss INStruction) indicate the nature of the requested operation (for example writing or reading). The following two (P1 P2) provide more information about the operation (for example an address). The response carries optional data bytes, and it ends with two status word SW1 and SW2.

2.3. Working with APDU

In a smart card, sometimes an entity named APDU Manager looks for the incoming ADPUs and performs the appropriate operations. For example, in a multi-application card, a *SELECT* APDU is used to choose a particular application (a piece of Java code, which is called a cardlet). Once an applet has been selected the APDU Manager sends all incoming APDUs towards this application.

In a terminal, a particular software driver manages the card reader. In 1996, the PC/SC standard [5] has been adopted by a set of computers and card manufacturers in order to support the integration of cards and readers in computer architecture. An application can send APDU to the smart card using APIs, which are made available by the host operating system.

We call AMUX (APDU Multiplexor) the layer located in the card or in the terminal that switches the APDU stream towards a card or a host application.

3. Using the classical smart card in a network

The traditional vision of the card can be seen in network applications, maybe because these devices are not specifically designed for such an environment [17,18]. As an example, a card is introduced in the security architecture working with SSL [16], thanks to software modifications (DLLs.) in the host system (including a web browser), and also in the server. The card knows nothing about network protocols, on the host side a specific piece of software plays with the card in order to authenticate its bearer by a challenge mechanism. An applet is downloaded from the server, gets control on the client machine, and is in charge of the communication between the card and the server.

In a more general way, the network applications (user authentication, electronic commerce.) use a *fat* client [13], each application dealing with a card requires specific software running on the terminal. Fat model means that the terminal machine (client side) needs to be reconfigured for each smart card type. This architecture generates difficulties of management if an application must be available through a set of heterogeneous terminals (PCs, Unix Hosts, GSM, Internet terminals …).

4. Smart card for networks

The fat client model does not meet the requirements needed by network applications, in which a mobile customer uses several terminals (laptop, mobile phone, Internet kiosk.) to access the Internet. For example, it seems obvious that an electronic subscription to a newspaper can be used at the home, office, hotel or from a mobile phone.

Our new approach to the network card consists of adapting it to each terminal by means of a unique protocol, which we wish to standardize. This protocol is, specifically, adapted to the support of the TCP/IP protocols. These new concepts are largely based on the Java Card technology, introduced in 1997 [14], which enables the realization of secure and open architectures (the smart card is not dedicated to a specific application).

The card implements the Internet protocols; it becomes a new object of the Internet community. The convergence of the world of traditional telephony and that of the networks and the availability of cheap Internet terminals are the factors that encourage the emergence of virtual objects (music, movies …), which can be bought and used through the network without a physical support. Our approach is a first step towards this type of model.

The most popular user interface is a web browser, this software implies that the terminal implements a TCP/IP stack, and has been fully configured for Internet purposes. The Internet card approach implies that the card is able to share the terminal network resources (which add no security features), such as the ethernet card and the TCP/IP stack.

4.1. Sharing the terminal TCP/IP stack

The card does not have the physical resources [19] for access to the network (ethernet card, modem …), the basic idea is to reach the software communication interfaces available on the host system.

In fact the same concept is used when a browser navigates, the application (the browser) is able to use the networks resources of the host system. The use of the network by an application is reduced to a cooperation with software layers provided by the host system, whose application knows an access point (or a method).

In the TCP/IP architecture, a network application (such as a web browser) is based on a model that comprises the first four layers of the OSI model [6] (Fig. 2).

Layers 1 (physical layer) and 2 (data link layer) represent, for example, the ethernet card or a modem. The machine reaches a network resource (ethernet card …) through a particular software interface sometimes named low-level driver (for example NDIS is a low-level driver developed by Microsoft). In fact, working with a network card means accessing to

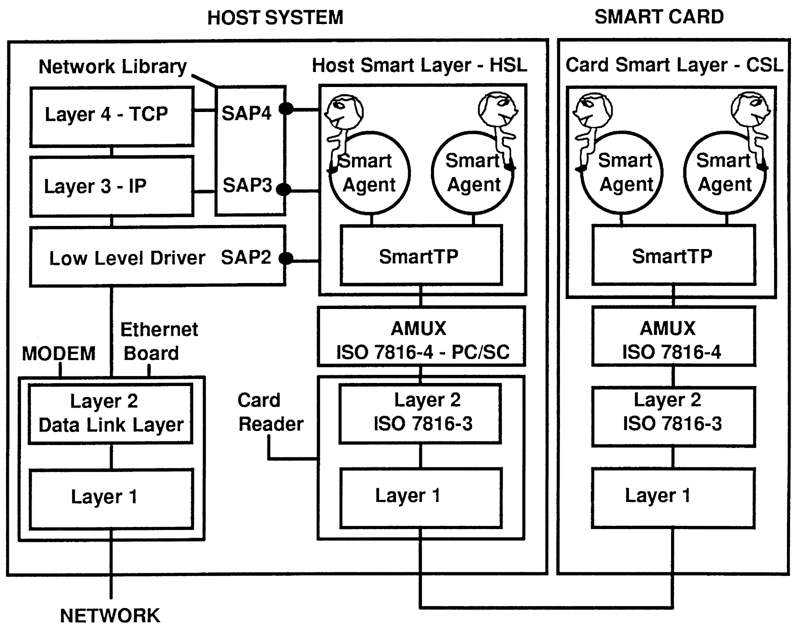

Fig. 2. Network card architecture.

a low-level driver, in ISO terminology this interface is named SAP (Service Access Point) of level 2 (SAP2)

In a similar way an application uses TCP/IP layers by means of a network library (for example the socket library) provided by the system; a network application can work with such a library through SAP of levels 3 (SAP3 or network layer SAP) or 4 (SAP4 or transport layer SAP). For example winsock.dll is the dynamic library, which makes the use of TCP/IP possible in a Windows machine.

5. Network card architecture

Our network card architecture is illustrated by Fig. 2. We have defined a new layer (Smart Layer), which uses the services of the AMUX entity. One layer is located in the host (HSL-Host Smart Layer) and the other in the card (CSL-Card Smart Layer).

The HSL layer has access to the network libraries and to the card reader APIs. It allows the transfer of the network packets from/to the card. It establishes a logical path between existing host applications, such as the web browser or electronic mail, and a smart card.

The CSL has access to the network by the means of information exchanged with HSL.

A smart layer is divided into two parts.

- Smart Agents.
- Smart Transfer Protocol entity (SmartTP).

A smart Agent is an Autonomous Software Entity (ASE). It can be materialized by a DLL (dynamic Link Library) in a PC, or by a Cardlet in a Smart Card. Smart Agents are identified by a reference (a 16 bit number), which can be either constant (a well-known value) or ephemeral. In the host side, Agents can reach the network resources; they provide an Internet access to Agents located in the card.

Three kinds of network Agents are defined according to the SAP, to which they are connected:

- Application network Agents, which are connected with a SAP4. In this case the whole terminal TCP/IP stack is used.
- Raw network Agents, which only work with a SAP3, and share the terminal IP layer.
- Packet network Agents, which play with a SAP2. For example they send or receive packet to/from an ethernet network.

Agents exchange information through packets that we call SmartTP pdu (SmartTP protocol data unit). SmartTP is a logical switch; it is in charge of routing the incoming pdu to Agents, and the outgoing pdu to the AMUX layer.

5.1. Network card stack

The communication stack (Fig. 3) used by a network card and its associated terminal is the following

- OSI layers 1 and 2, supporting the ISO 7816-3 transmission protocol.
- AMUX layer, which routes the APDU to the SmartTP entity.
- SmartTP entity, which switches the SmartTP pdu toward Agents.
- Agents, which process application data, and which receive and transmit SmartTP pdu.

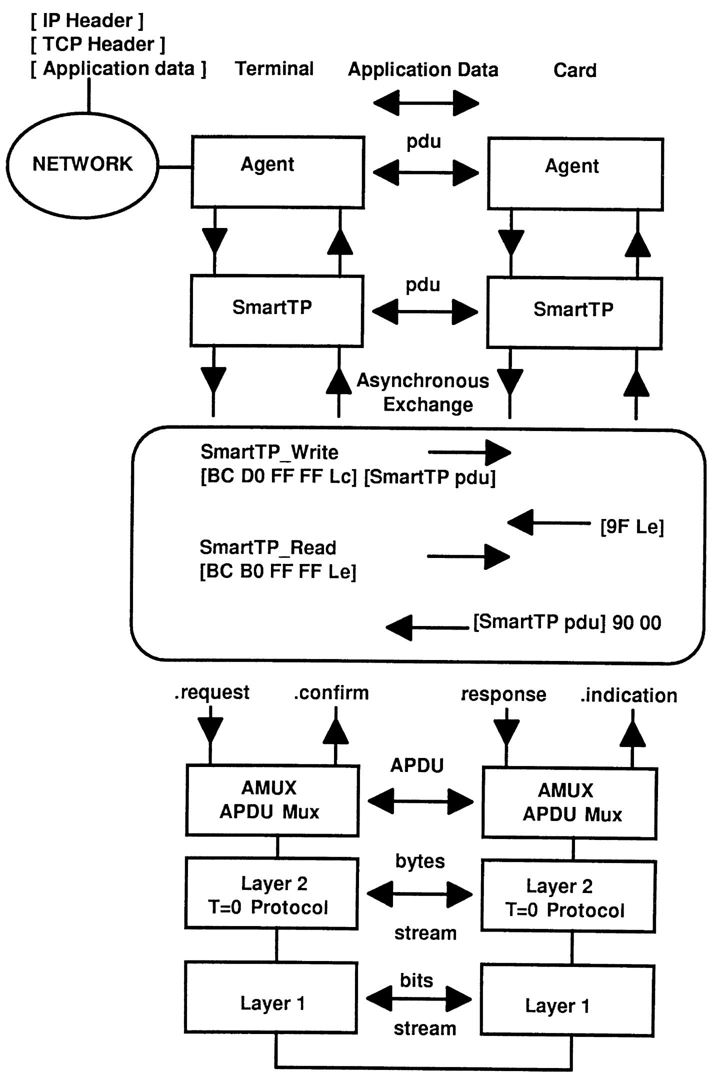

Fig. 3. Network card stack.

5.2. ISO 7616-3 transmission protocol T = 0

This protocol is the most widely used to exchange APDU between the card and the terminal. A serial link working with 8 bit words, start, stop and parity bits, carries exchanged bytes.

5.3. AMUX interface

Two specific APDUs (Fig. 3), *SmartTP_Write* and *SmartTP_Read* have been defined in order to carry SmartTP pdu by using the T = 0 transmission protocol services. According to the OSI concepts, the logical interface with the SmartTP entity is a set of four primitives, AMUX.request, AMUX.indication, AMUX.response, AMUX.confirm.

5.4. SmartTP protocol

The SmartTP protocol looks like a *TCP light* protocol. In the TCP protocol [7] port numbers are used to identify the software entities that are connected. Six flags (URG, ACK, PSH, RST, SYN, and FIN) are used to manage a session. Because data are received in order and free from error, SmartTP only deals with the port and flag attributes

A SmartTP pdu (Table 1) is divided into four parts

- *Source Reference* (2 bytes), the reference of the Agent which has issued this pdu.
- *Destination Reference* (2 bytes), the reference of the Agent to which the pdu is sent.

Table 1
SmartTP protocol data unit

Source reference 2 bytes	Destination reference 2 bytes	Flags 1 byte	Optional data

- *Flags* (1 byte).
- *Data*, information (0–240 bytes) exchanged by Agents, this field is optional.

Flag is a set of 8 bit indicators, three of them are mandatory.

- *Open*, when set, this indicator means that a session is being opened between the destination and the source Agent.
- *Close*, when set, this indicator means that a session has been closed between the destination and the source Agent.
- *Block*, when set this indicator notifies to the destination Agent that the source Agent is waiting for a response and remains in a freezed state.

Others indicators (like *Read* and *Write*) are helpful for many Agents; five indicators (Open, Close, Block, Read, Write) are sufficient to play with the well-known paradigm (OPEN READ WRITE CLOSE), which is used in UNIX to manipulate files. We shall represent a pdu by the following notation:

- [s = SourceReference,d = DestinationReference], a token with no indicator set.
- [s = SourceReference,d = DestinationReference, Block + Write], a token whose indicators Block and Write are set.
- [s = SourceReference,d = DestinationReference, Open, Data], a pdu whose open indicator is set and which includes data.

5.5. SmartTP entity

SmartTP is in charge of switching packets to the destination Agent. It is associated with a Null reference. A Token is a pdu with no data; a Null Token is issued by SmartTP, whose source reference is null. On the contrary a Data pdu includes information bytes.

The main SmartTP rules are the following:

1. SmartTP (smart card side) always sends a pdu following an incoming pdu; it can be either an Agent pdu or a Null Token.
2. SmartTP (terminal side) sends nothing upon reception of a Null token.
3. SmartTP sends a Null Token, if a pdu is received without the Open indicator set, with an unknown destination reference.
4. SmartTP produces a token with a Close indicator set, upon the reception of an Open pdu whose destination reference is unknown.

5.6. Agents

Two Agents are connected via a session. A given Agent is identified by an integer (ranging between 0 and 65535, 16 bits). The most significant bit (b15) of the reference indicates as to whether an Agent is local (located in the same smart layer) or remote (b15 = 0). This means that Agents can communicate within a smart layer or between two smart layers (terminal side and card side)

There are six specific properties characterizing Agents:

- *Terminal*, an Agent located in the host system.
- *Card*, an agent located in the card.
- *Local*, an Agent that cannot use the network resource.
- *Network*, an Agent that can access the network.
- *Client*, an Agent that intializes a session (via the Open indicator).
- *Server*, an Agent that receives a request for opening a session (Open).

In the terminal side, network Agents are able to access the network resources (low-level driver, network library). They are the key components of network cards, from a TCP/IP point of view these Agents are acting as a server or client; they give an Internet access to the card Agents to which they are connected.

5.7. Agents state

An Agent is associated with three basic states:

- *Disconnected*, no session is opened with another Agent. When an Agent sends or receives an Open pdu it switches to the blocking state
- *Waiting and Connected*, a session is available with other Agents. Agent is ready to process data towards/from the network/from another Agent. In this state, the Agent can switch to the Blocking state (data is ready for the remote session Agent) or to the disconnected state (a Close pdu has been sent or received).
- *Blocking*, in this state an Agent is waiting for an incoming pdu. Upon reception of this pdu it switches to the waiting state or to the disconnected state (if a Close indicator has been received).

5.8. Agent session

A session is a logical link between two Agents. Agents are identified by a reference. A client sends a pdu for which the OPEN indicator is set (request for session opening). The reference of a client is ephemeral, i.e. it is allocated by SmartTP by means of the GiveReference() function. SmartTP determines an ephemeral reference, for example by according to an LRU algorithm (Last Recently Used). A server receives a request for a session opening. The reference of a server is constant; i.e. it is a well-known value. A session is identified by a couple, client reference and server reference. The ephemeral client reference enables this couple to be pseudo unique (for a reasonable amount of time). The transmission of a pdu with the indicator

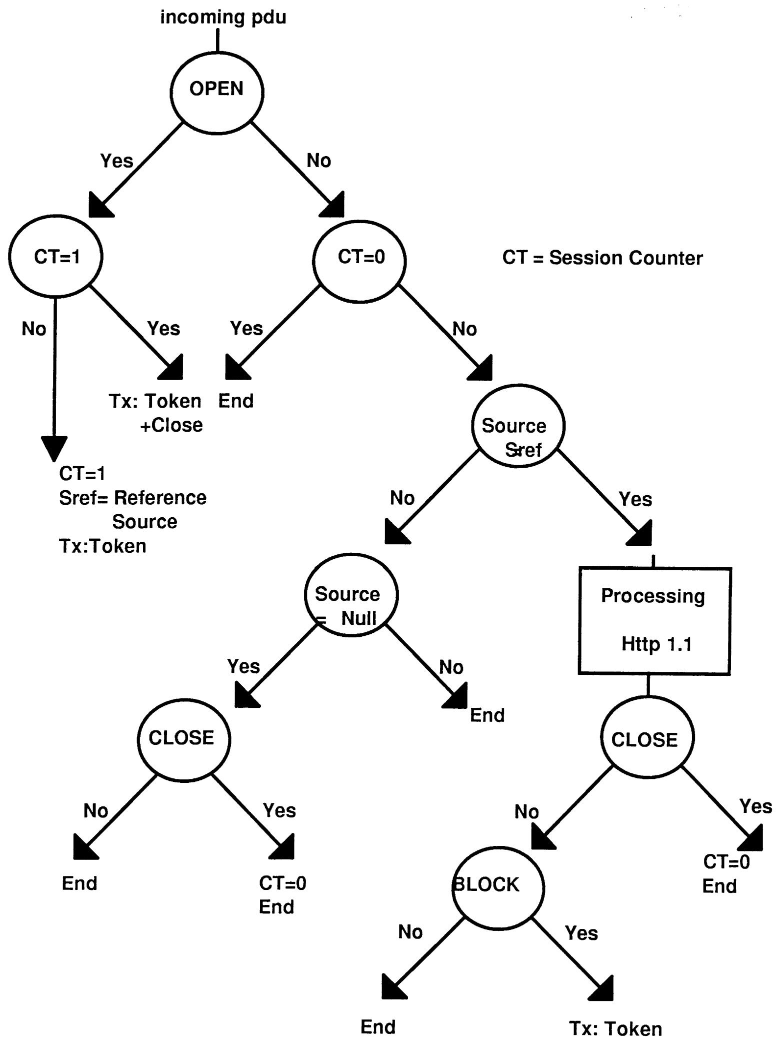

Fig. 4. Server Agent state machine.

CLOSE ends a session. A client notifies SmartTP that a session has been ended by means of the function Release Reference().

5.9. Rules for Agents

1. An Agent ignores (no outgoing pdu is generated) each pdu whose open indicator is not set, and which cannot be associated to an open session.
2. An Agent never responds (no outgoing pdu is generated) to a pdu whose close flag is set. If necessary a token will be generated by SmartTP.
3. Upon reception of a pdu whose source reference is null and with the close indicator, an Agent closes all its open sessions and no outgoing pdu is generated.
4. An Agent always responds to a pdu whose blocking indicator is set, if necessary by a token.

Fig. 4 illustrates the implementation of these rules in a card server Agent. The processing state is, for example, in charge of performing the HTTP 1.1 protocol.

5.10. Network Agent

The network Agent is the key part of a network card. It acts as a proxy between a TCP/IP connection and a (card) remote Agent. Due to the intrinsic properties of TCP protocol, two kinds of network Agents are defined: TCP server and TCP client. A network Agent transforms incoming TCP connection in outgoing SmartTP open pdu, and incoming Open SmartTP pdu in outgoing TCP connection. Once a TCP connection is established and a session to a remote Agent is opened, data is relayed between the remote Agent and the TCP connection; and a network agent creates a tunnel between a TCP connection and a land agent. A network Agent acts as a protocol converter between TCP and SmartTP. We will now describe the network Agent that we have developed (Fig. 5).

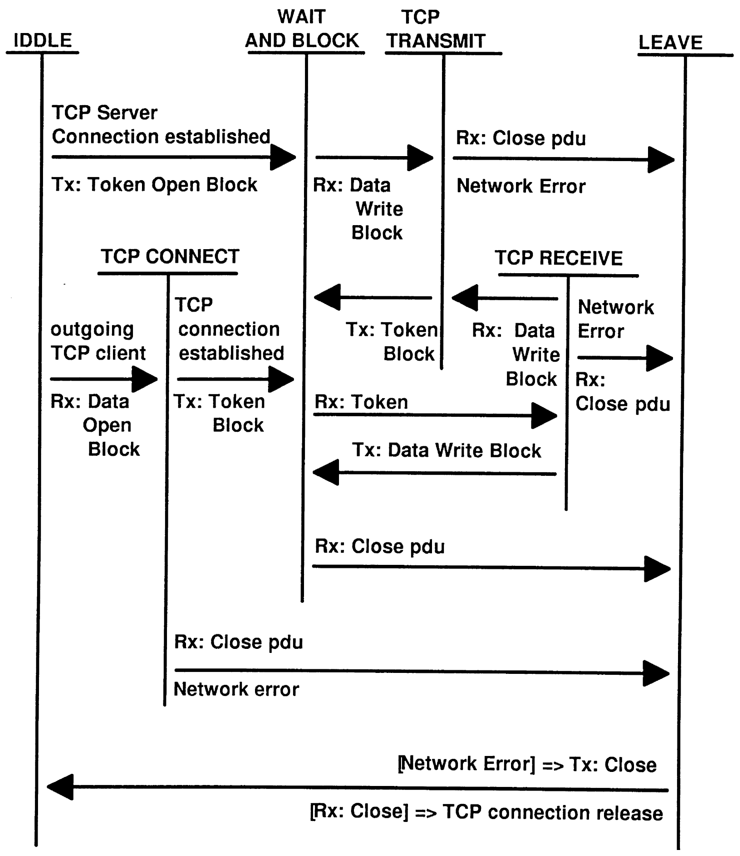

Fig. 5. Network Agent state machine.

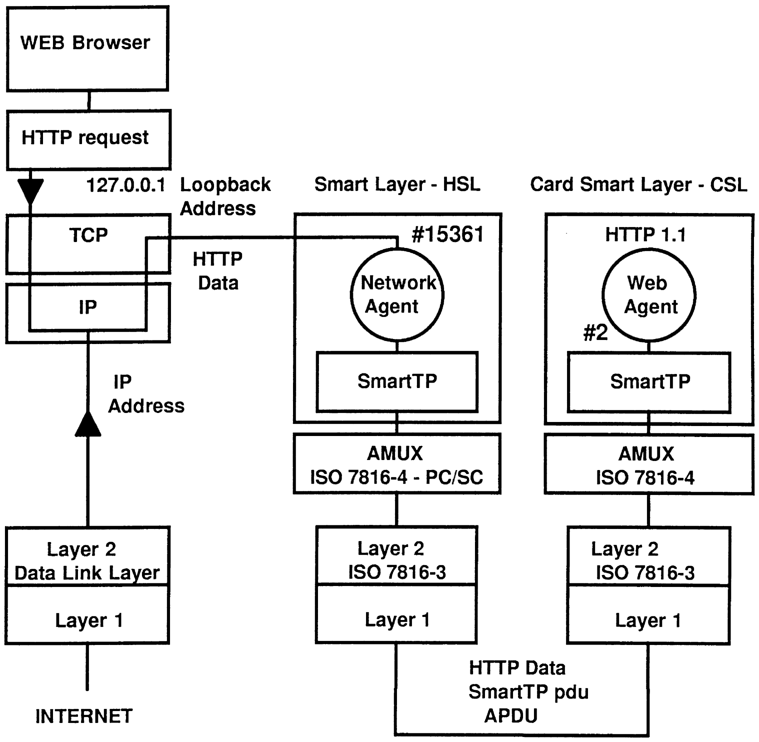

Fig. 6. Smart card as a web server.

5.11. Network Agent state machine

Our network Agent is organized around a central state WAIT_AND_BLOCK, in which no operation is performed. The system is waiting for an acknowledgment (a token), which will specify the next Agent state (writing to the network or reading from the network). This Agent includes six states.

- IDDLE, in this state a TCP server (SmartTP client) is waiting for an incoming TCP connection, or a SmartTP server is waiting for an incoming OPEN pdu.
- TCP_CONNECT, this state is meaningful only for a TCP client. An incoming *open pdu* has been received. The associated data specified the host name (or IP address) and the TCP port. A TCP connection is in progress.
- WAIT_AND_BLOCK, in this state the Agent is waiting for an incoming pdu; no network operation is in progress.
- TCP_TRANSMIT, data are being sent through the network.
- TCP_RECEIVE, the Agent is waiting for data from the network.
- LEAVE, if a network error has occurred a *close token* is sent, otherwise a *close pdu* has been received and the TCP connection is closed.

6. Smart card as a web server

A web server is an Internet protocol specified by an RFC standard (RFC 2068-HTTP 1.1 [10]). Its implementation in a card is such that the http data, which are carried through the web by TCP/IP packets, are exchanged between the card and the terminal by means of SmartTP pdu. From the application point of view an http session is opened between the client (web browser) and a web server located in the card.

A traditional card is APDU centric; each card resource is available through an APDU. A network smart card is url centric; every resource can be reached by an url (an http request).

Let us consider a terminal with a web browser. The url *http://127.0.0.1:8080,* where *127.0.0.1* is the terminal IP loop back address and *8080* the TCP port of the network Agent, gives access to the card general index (for example an html file named index.html). This page includes hyperlinks towards internal or external resources associated to the card. We call *Virtual Terminal* the fact that through its web server, a card is able to exhibit a virtual representation of its resources. A card resource can be a cryptographic entity (cryptographic algorithm, digital signature, authentication procedure), a multimedia object (html page, image, sound…) or a piece of software (java applet…). An Internet card is an open device, which can be used without any particular knowledge of its operating or file systems.

Because a TCP/IP terminal has both a loop back IP address (127.0.0.1), and a usual IP address, a network card can be accessed from a terminal or from any web node. A practical constraint for a network Agent design is that a web browser opens simultaneously several TCP connections (four is the usual value). Therefore, it is necessary to serialize (for example by means of a semaphore) the processing of these incoming sessions.

6.1. An http session

An http session runs in two phases: first a TCP client sends a request to a web server; and second the web server replies by a header (which defines the file content) followed by the requested file. In http protocol a header precedes a file, so the card operating system must be able to deliver this specific element for each file. We now give more details about an http session (Fig. 6)

- A new connection occurs between a TCP client (web browser) and the TCP server associated to a network Agent. This Agent gets an ephemeral reference from SmartTP (for example #15361) and sends an open + block token to a well-known remote (card) Agent (a web server whose reference is equal to #2). A session is opened between terminal Agent #15361 and card Agent #2.
- The web Agent sends a token.
- Data received (by Agent #15361) from the TCP client (http request) are segmented and sent to the web Agent (#2) through write + block pdu. Each pdu is acknowledged by a token.
- At the end of the http request, the file and its associated header are segmented and sent to the network Agent. The web Agent sends write + block pdu to the network Agent.
- The network client sends data carried in this pdu to the TCP client over the TCP connection. The network Agent produces a token upon the completion of each TCP transmission.
- At the file end, the web Agent sends the last pdu with the close indicator set. Upon its reception the network Agent releases the TCP connection.

6.2. PDU exchange example

- A web browser opens a session with a network Agent.
  [s = 15361,d = 2,open + block], [s = 2,d = 15361].
- An http request is sent from a browser towards the card web server.
  [s = 15361,d = 2,write + block,data],
  [s = 2,d = 15361].
- Last part of the request is sent
  [s = 15361,d = 2,write + block,data].
- The card server sends its response.
  [s = 2,d = 15361,write + block,data],
  [s = 15361,d = 2,block].
  [s = 2,d = 15361,write + block,data],
  [s = 15361,d = 2,block].

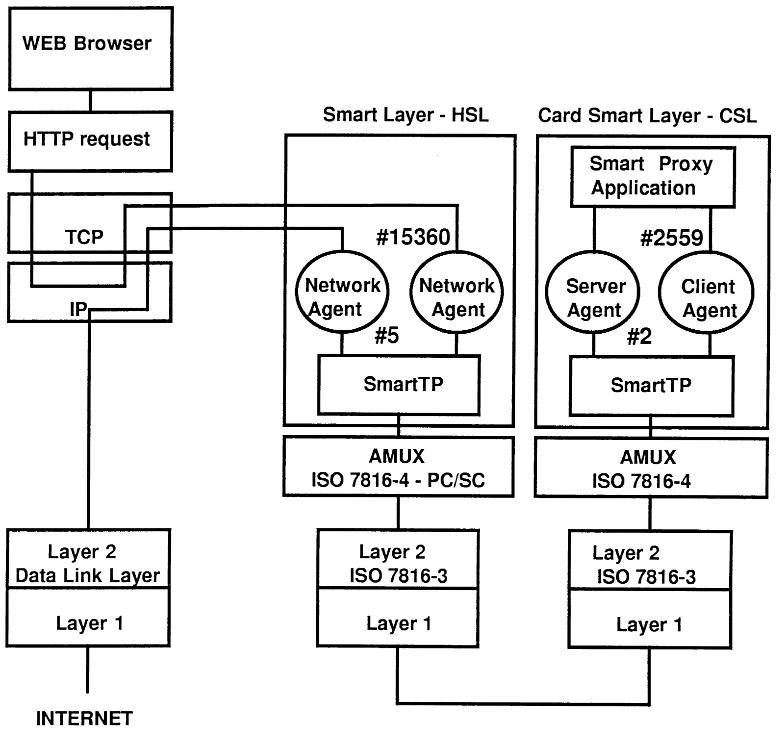

Fig. 7. Smart card as an Internet proxy.

- The card server closes the TCP connection.
  [s = 2,d = 15361,write + close,data].

7. Smart card as an Internet proxy

TCP proxy is a very powerful and useful entity in the world of TCP/IP technology. It includes a static TCP server and a TCP client, which is created dynamically upon each new incoming connection to the server. The client can be connected to a pre-defined Internet node, or to a node that is determined from the information received through the server connection. A proxy forwards the application data (carried by TCP/IP packets) from one TCP connection to another. It can be used for security purposes (http proxy, firewall), as a gateway between the Intranet and Internet; it can provide protocol translation etc.

A smart card proxy replaces a card url by a particular procedure, which implies a connection to a server located somewhere in the web. The connection procedure, which can include authentication features, is managed by software running in the card. The card bearer knows nothing about the cryptographic algorithms and their associated keys. For example, an url like http://127.0.0.1:8080/eMail can be used to reach an eMail server, the smart card will interpret this request by a connection to a particular server and will be in charge of the authentication process.

A file that is identified by a card url (see above), but which is stored elsewhere in a remote server is called a *virtual file*.

More generally, smart card proxy is a basic element for a new generation of applications, which will use the card as a trusted proxy. The card acts as an active filter between a user and the network. Incoming and outgoing data can be processed or stored in the card, according to predefined rules.

A smart card proxy (Fig. 7) forwards data between two network Agents by means of two simultaneous sessions. Data coming from the network are processed by an application located in the card. For example, an SSL proxy located in a smart card can totally handle a secure session; it checks the server certificate, a vulnerable point of the SSL procedure if the terminal is not trusted.

7.1. A proxy session

A proxy session works in several steps:

1. A connection is established with a card server (for example a web server), this server is reached through a network Agent (#15360)–smart card Agent (#2) session.
2. Data are exchanged between the network client and the card server. From these data, the proxy application located in the card determines a remote client address.
3. Card application establishes a connection to a remote Internet server. A session is opened between a card client Agent and a network Agent. Card Agent gets an ephemeral reference (#2559) and sends an open + block pdu to a network Agent (whose reference is well known, for example #5). Data associated with this pdu specify a

port and server address. The network Agent (#5) initiates a TCP connection. Upon success a token is sent to card Agent #2559.

4. Card Agent #2559 exchanges data with the Internet server by means of the session opened with network Agent #5.
5. Data received from the network Agent #5 are forwarded (through the card proxy application) to network Agent #15360 and vice versa.
6. A network Agent closes its associated session due to the fact that a TCP connection has been released.

7.2. PDU exchange example

- A first session is opened, between a browser, and the smart card web server
  [s = 15360,d = 2,open + block], [s = 2,d = 15360].
- The browser sends an http request to the card web server.
  [s = 15360,d = 2,write + block,data],
  [s = 2,d = 15360,ack].
- Last part of this request is sent.
  [s = 15360,d = 2,write + block,data].
- Smart card proxy opens a second connection towards a remote server.
  [s = 2559,d = 5,open + block,data],
  [s = 5,d = 2559,block].
- Smart card proxy sends data (an http request) to this remote server.
  [s = 2559,d = 5,write + block,data],
  [s = 5,d = 2559,block].
- Last part of the request is sent.
  [s = 2559,d = 5,write,data].
- Smart card proxy forwards data between the two terminal network Agents.
  [s = 5,d = 2559,block], [s = 2,d = 15360,block],
  [s = 15360,d = 2], [s = 2559,d = 5].
- Data are received from the network and sent through the card to the browser.
  [s = 5,d = 2559,write + block,data],
  [s = 2,d = 15360,write + block,data].
  [s = 15360,d = 2,block], [s = 2559,d = 5,block]
  [s = 5,d = 2559] [s = 2,d = 15360].
  Agent #5 is ready to receive data again.
- Session end.
  The card proxy forwards data …
  [s = 5,d = 2599,write + block],
  [s = 2,d = 15360,write + block].
  Agent #15360 releases a (blocking) token.
  [s = 15360,d = 2,block], [s = 2559,d = 5,block].
  The web browser ends the TCP connection sessions, Agent #15360 closes its session.
  [s = 15360,d = 2,close], [s = 2559,d = 5,close].
  Agent #5 close its session with Agent #2559, but two pdus, which were produced before the session end, are sent by HSL to CSL. CSL acknowledges each of them by a Null Token.
  [s = 5,d = 2559], [s = 0,d = 0].
  [s = 5,d = 2559,write + block,data], [s = 0,d = 0].

8. Conclusion

We have implemented the Internet card concept in a Java card whose user memory size is 7 kbytes. The java code size which includes a web server, a smart proxy and a file system is about 3.5 kbytes; 3.5 kbytes are available for file contents. Html pages, images and applets have been stored in the card and can be accessed through the web server. It is possible to protect some smart card files by a lock, each time a web browser tries to access a locked file an html form is produced, which notifies the user to specify a password.

For demonstration purposes a smart proxy transforms an http request to a file named eMail. This is an outgoing connection to a free eMail server to which the card sends a login and a password encapsulated in a form. The result is that a user can access an eMail server without knowing the server location and the needed password.

HSL has been written for a win32 system, the code size is small, around 100 kbytes.

Measured performance, from a data rate point of view is between 100 and 200 bytes/second (half for proxy application), according to the type of reader used.

We have demonstrated that it is possible to use a smart card as an Internet node. We think that this technology is a first step towards new network applications dealing with *virtual objects*. Smart cards working with 32 RISC processors will appear in early 2000. These devices should be able to support the higher data rates that are needed for applications, such as dealing with multimedia objects and requiring security features (authentication, integrity, and privacy).

References

[1] International Organization for Standardization, Identification Cards-Integrated Circuits(s) Cards with Contacts, ISO 7816.
[2] International Organization for Standardization, Contactless integrated circuit(s) cards-Proximity Cards, ISO 14443.
[3] European Telecommunication Standards Institute, Digital cellular telecommunications system (Phase 2 +) Specification of the Subscriber Identity Module–Mobile Equipment (SIM-ME) interface ETSI GSM 11.11.
[4] European Telecommunications Standards Institute, Digital cellular telecommunications system (Phase 2 +) Specification of the SIM Application Toolkit for the Subscriber Identity Module-Mobile Equipment (SIM-ME) interface, ETSI GSM 11.14.
[5] Interoperability Specification for ICCs and Personal Computer Systems, PC/SC, © 1996 CP8 Transac, HP, Microsoft, Schlumberger, Siemens Nixdorf.
[6] International Organization for Standardization, Information Processing Systems-Open Systems Interconnection-Basic Reference Model, ISO 7498.

[7] J. Postel, Transmission Control Protocol, Request For Comment RFC 793, September 1981.
[8] L.C. Guillou, M. Ugon, J.-J. Quisquater, The smart card: a standardized security device dedicated to public cryptology, in: G.J. Simmons (Ed.), Contemporary Cryptology. The Science of Information Integrity, IEEE Press, 1992, pp. 561–613.
[9] R. Merckling, A. Anderson, Smart Card Introduction, Request For Comment RFC 57, March 1994.
[10] T. Berners-Lee et al., Hypertext Transfer Protocol — HTTP/1.1, Request For Comment, RFC 2068, January 1997.
[11] R.W. Baldwin, C.V. Chang, Locking the e-safe, IEEE Spectrum (February) (1997) 40–46.
[12] C.H. Fancher, In your pocket: smartcards, IEEE Spectrum (February) (1997) 47–53.
[13] C. McChesney, Banking in cyberspace: an investment in itself, IEEE Spectrum (February) (1997) 54–59.
[14] C. Markantonakis, Information Security Workshop 97 (ISW'97), September 1997 (Ishikawa in Japan), The Case for a Secure Multi-Application Smart Card Operating System, Springer (LNCS 1396), Berlin, 1997, pp. 188–197.
[15] R. Dettmer, Getting smarter, IEE Review (May) (1998) 123–126.
[16] T. Verschuren, Smart access: strong authentication on the web, Computer Networks and ISDN Systems 30 (1998) 1511–1519.
[17] N. Itoi, P. Honeyman, Smartcard integration with Kerberos V5, Proceedings of USENIX workshop on Smartcard Technology, Chicago, May 1999, pp 51–62.
[18] N. Itoi, P. Honeyman, J. Rees, SCFS: a UNIX Filesystem for Smartcards, Proceedings of USENIX Workshop on Smart Card Technology, Chicago, May 1999, pp 107–118.
[19] B. Schneier, A. Shostack, Breaking Up Is Hard to Do: Modeling Security Threats for Smart Cards, Proceedings of USENIX Workshop on Smart Card Technology, Chicago, May 1999, pp 175–185.
[20] P. Urien, H. Saleh, A new network smart card approach, JRES99 3° journées réseaux Ministère de l'éducation nationale, de la recherche et de la technologie, December 1999.
[21] J.P. Tual, MASSC, a generic architecture for multi-application smart cards, IEEE Micro (September–October) (1999).
[22] K. Vedder, F. Weikmann, Requirements, Properties and Applications (Chipkarten Grundlagen, Realisierungen, Sicherheitsaspekte, Anwendungen), Verlag Vieweg, 1998, pp. 2–23 (ISBN 3-528-05667-3).
[23] R. Anderson, M. Kuhn, Tamper Resistance — a Cautionary Note, Second Usenix Workshop on Electronic Proceedings, Oakland, California, November 18–21, 1996, pp 1–11, ISBN 1-880446-83-9.
[24] http://www.cryptography.com/dpa
[25] Information Technology Security Evaluation Criteria (ITSEC), Harmonised Criteria of France, Germany, the Netherlands, United Kingdom, V1.2, European Commission Luxembourg, June 1991.

Pascal Urien is currently in charge of research on smartcard integration in networks, at the Bull Smart Cards and Terminals R&D Division. He is also teaching networks and IP technologies at the French University of Paris Dauphine. Pascal graduated from French Ecole Centrale Lyon; he wrote a thesis in solid state physics, and received a PhD in Computer Science.